

Testes e implantação de aplicações

Geordano Lima

Introdução aos Testes de Software



O que é um teste de software?

Simplificando, é um processo que visa verificar se um software funciona conforme o esperado e identificar possíveis erros ou defeitos.



Por que testar?

- Qualidade
- Confiabilidade
- Manutenção
- Economia



E se não testar?



Ônibus espacial Challenger

A explosão do ônibus espacial Challenger em 1986 foi causada por uma falha em uma junta de vedação, que não foi detectada em testes de sistema adequados. Essa tragédia custou a vida de sete astronautas e atrasou o programa espacial por anos.

- Imagem da explosão do ônibus espacial

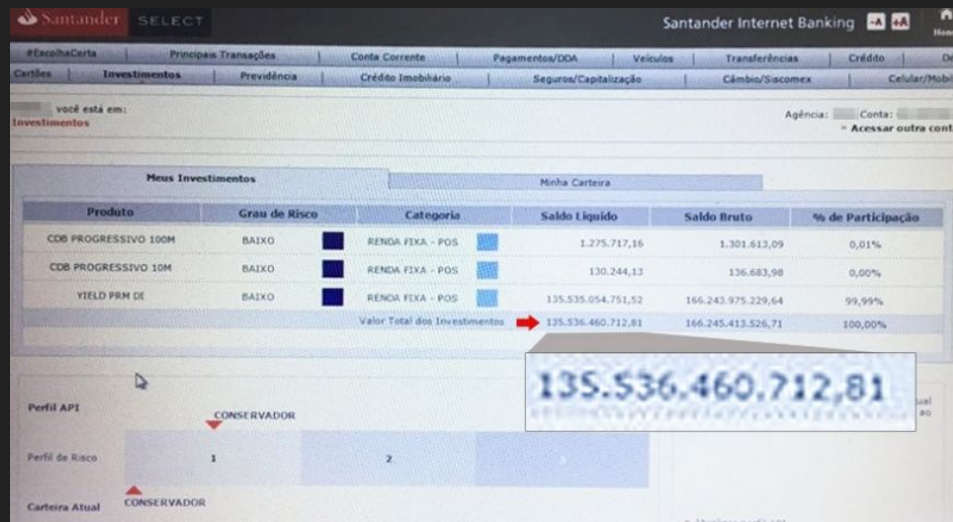


Fonte: <https://olhardigital.com.br/2023/01/27/ciencia-e-espaco/onibus-espacial-challenger-tragedia-que-chocou-o-mundo-completa-37-anos/>

Falhas em sistemas bancários

Erros em sistemas bancários podem levar a perdas financeiras significativas para os clientes e instituições. Por exemplo, sistemas de pagamento que não são devidamente testados podem processar transações incorretas, causando prejuízos e afetando a reputação do banco.

- Imagem de uma falha em um processo do banco em 2017, que enviou para conta de uma pessoa milhões de reais



The screenshot shows the Santander Internet Banking interface. The 'Meus Investimentos' (My Investments) section is active, displaying a table of investments. The table has columns for Product, Risk Level, Category, Liquid Balance, Gross Balance, and % Participation. The 'Valor Total dos Investimentos' (Total Value of Investments) is highlighted with a red arrow and a callout box showing the value 135.536.460.712,81. Below the table, the 'Perfil API' (API Profile) is shown as 'CONSERVADOR' (Conservative), and the 'Carteira Atual' (Current Portfolio) is also labeled 'CONSERVADOR'.

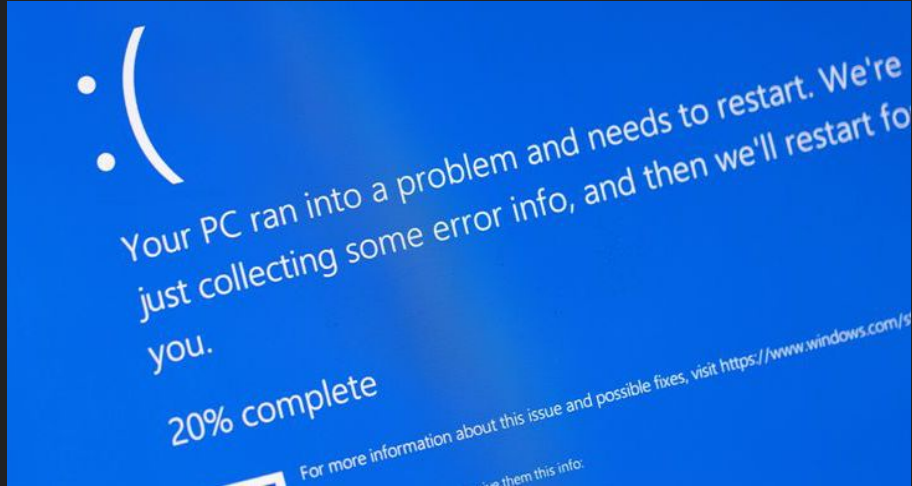
Produto	Grau de Risco	Categoria	Saldo Líquido	Saldo Bruto	% de Participação
CDB PROGRESSIVO 100M	BAIXO	RENTA FIXA - POS	1.275.717,16	1.301.613,09	0,01%
CDB PROGRESSIVO 10M	BAIXO	RENTA FIXA - POS	130.244,13	136.683,98	0,00%
YIELD PRM DE	BAIXO	RENTA FIXA - POS	135.535.054.751,52	166.243.975.229,64	99,99%
Valor Total dos Investimentos			135.536.460.712,81	166.245.413.526,71	100,00%

Fonte: <https://tecnoblog.net/noticias/erro-banco-bilhoes/>

Lançamentos de software com bugs críticos

A falta de testes de sistema pode levar ao lançamento de software com bugs graves que comprometem a funcionalidade e a segurança do sistema. Isso pode resultar em perda de dados, falhas de segurança e insatisfação do cliente.

- Imagem - Erro da CrowdStrike em 2024 que causou problemas nos sistemas operacionais Windows.



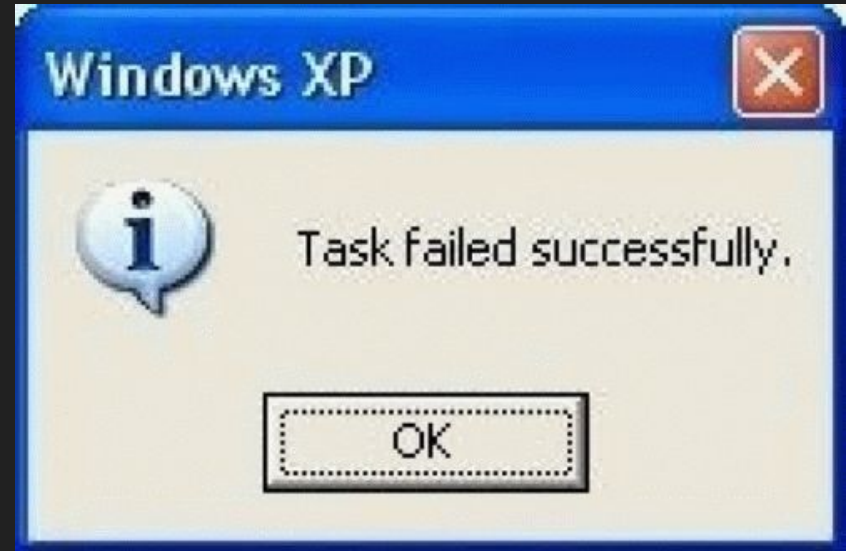
Fonte: <https://tecmania.com.br/crowdstrike-esclarece-as-causas-de-falha-massiva-em-dispositivos-windows/>

Mais exemplos

Falhas em sistemas de controle de tráfego aéreo: Um sistema de controle de tráfego aéreo com falhas pode levar a colisões aéreas e outras catástrofes.

Falhas em sistemas de segurança: Sistemas de segurança mal testados podem permitir invasões e roubos, causando prejuízos financeiros e danos à reputação.

Falhas em sistemas de energia: Sistemas de energia que não são devidamente testados podem causar apagões em larga escala, afetando milhões de pessoas e empresas.



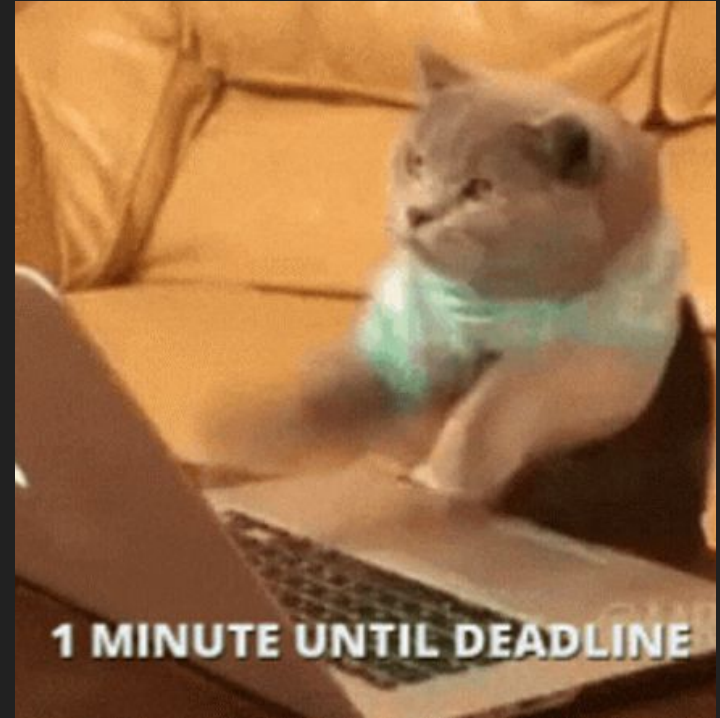
Mas por que isso ocorre?

Pressão por prazos: A pressão por lançar produtos rapidamente pode levar a cortes nos testes.

Custos: Os testes podem ser caros e demorados, levando algumas empresas a economizar nessa etapa.

Complexidade: Sistemas complexos podem ser difíceis de testar completamente.

Falta de experiência: Equipes com pouca experiência em testes podem não realizar testes adequados.



Quais são os tipos de testes?

- Testes de unidade (unitário)
- Testes de integração
- Testes de componentes
- Testes de sistema (e2e)
- Testes de aceitação
- Testes de desempenho
- Testes de segurança



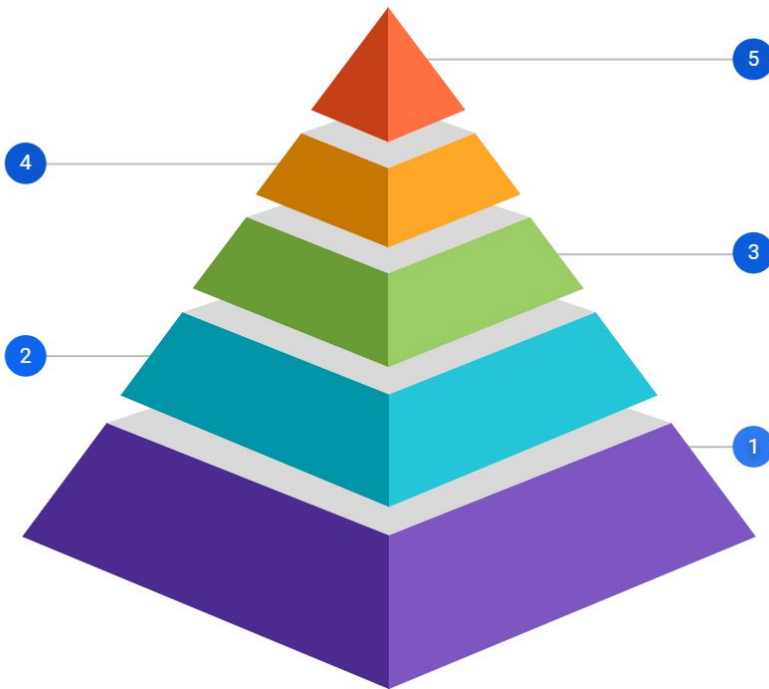
Pirâmide de testes

Ponta-a-ponta (e2e)

- Que segue a jornada do usuário do início ao fim.

Integração

É a fase do teste onde os módulos são combinados e testados em grupo.



Exploratório

- Onde realizamos os testes manuais e exploratórios com o objetivo de criticar o produto e executá-lo como um usuário real.

Componente

- Aqui, testamos os componentes isolados e verificamos as interações entre suas classes.

Unitário

- É a base da pirâmide, eles são rápidos, baratos e sem dependências. Onde cada unidade do sistema é testada individualmente.

Testes de unidade (unitário) - unit tests

```
import pytest

def soma(valor1: float, valor2: float) -> float:
    return valor1 + valor2

def test_soma():
    valor1 = 2
    valor2 = 3
    resultado = soma(valor1=valor1, valor2=valor2)
    assert resultado == 5

def test_soma_incorreta():
    valor1 = 2
    valor2 = 2
    resultado = soma(valor1=valor1, valor2=valor2)
    with pytest.raises(AssertionError):
        assert resultado == 5
```

Exercício:

Desenvolva uma calculadora

- Deve receber 2 valores e a operação
 - operações:
 - "+" soma
 - "-" subtração
 - "/" divisão
 - "*" multiplicação
- Deve fazer um teste para cada operação
- Deve fazer um teste de erro

Sugestão de chamada de método utilizando python:

```
calcular(operacao="+", valor_1="10", valor_2="15")
```

> 25

Estruturas de programação

- estruturas de controle

- if
- elif
- else

```
nota = 75

if nota >= 70:
    print("Aprovado!")
elif nota >= 60:
    print("Recuperação!")
else:
    print("Reprovado!")
```

> Aprovado

- estruturas de loop:

- for
- while

```
for i in range(5):
    print(i)
```

> 0
> 1
> 2
> 3
> 4

```
contador = 0
while contador < 5:
    print(contador)
    contador += 1
```

> 0
> 1
> 2
> 3
> 4

Estruturas de programação

- Funções
 - função sem retorno
 - função com retorno

```
def saudacao(nome):  
    print(f"Olá, {nome}!")  
  
saudacao("Maria")
```

> Olá, Maria!

```
def calcular_area_retangulo(base, altura):  
    area = base * altura  
    return area  
  
resultado = calcular_area_retangulo(5, 3)  
print(resultado)
```

> 15

Estruturas de programação

- Listas
 - selecionar item da lista
 - adicionar item na lista
 - remover item da lista

```
frutas = ["maçã", "banana", "laranja"]  
print(frutas[0])  
frutas.append("uva")  
frutas.remove("banana")  
print(frutas)
```

```
> maçã  
> ['maçã', 'laranja', 'uva']
```

Estruturas de programação

- Tuplas
 - criando tuplas
 - acessando elementos da tupla

```
coordenadas = (10, 20)  
print(coordenadas[0])  
print(coordenadas)
```

> 10

> (10, 20)

Estruturas de programação

- Dicionário
 - criando dicionário
 - acessando valores
 - criando novos valores
 - alterando valores
 - removendo valores

```
aluno = {"nome": "João", "idade": 20, "notas": [8, 7, 9]}
print(aluno.get("notas", [0, 0, 0]))
print(aluno.get("matricula", 0))
aluno["idade"] = 21
print(aluno["idade"])
aluno["sobrenome"] = "Teste"
print(aluno["sobrenome"])
print(aluno)
aluno.pop("sobrenome")
print(aluno)
```

> [8, 7, 9]

> 0

> 21

> Teste

> {'nome': 'João', 'idade': 21, 'notas': [8, 7, 9], 'sobrenome': 'Teste'}

> {'nome': 'João', 'idade': 21, 'notas': [8, 7, 9]}

Estruturas de programação

- Classe
 - criando uma classe
 - criando uma ação para classe
 - instanciar a classe
 - buscar informações da classe
 - executar a ação da classe

```
class Cachorro:
    def __init__(self, nome, raca):
        self.nome = nome
        self.raca = raca

    def latir(self):
        print("Au au!")

meu_cachorro = Cachorro("Rex", "Labrador")
print(meu_cachorro.nome, meu_cachorro.raca)
meu_cachorro.latir()
```

> Rex Labrador

> Au au!

Estruturas de programação

- Classe
 - criando uma classe
 - criando uma ação para classe
 - instanciar a classe
 - buscar informações da classe
 - executar a ação da classe

```
try:  
    resultado = 10 / 0  
except ZeroDivisionError:  
    print("Não é possível dividir por zero.")
```

> Não é possível dividir por zero.

Estruturas de programação

- Arquivos
 - criar um arquivo
 - escrever dentro do arquivo
 - acessar arquivo
 - ler conteúdo do arquivo

```
arquivo = open("meu_arquivo.txt", "w")  
arquivo.write("Olá, mundo!")  
arquivo.close()
```

```
arquivo = open("meu_arquivo.txt", "r")  
conteudo = arquivo.read()  
print(conteudo)  
arquivo.close()
```

> Olá, mundo!